FLAME: A Flow-level Anomaly Modeling Engine

CSET Workshop 2008
Daniela Brauckhoff*, Arno Wagner, Martin May
ETH Zurich, Switzerland
Motivation

- Many approaches for NetFlow-based anomaly detection developed in recent years
  - PCA, Kalman/bloom filter, clustering, wavelets, SOMs, sketches,...

- Evaluation of approaches is difficult since appropriate evaluation traces are not available
  - Appropriate means labeled, versatile, representative, and of sufficient size and length
Available Evaluation Methods

- **Trace Merging**
  - Merge captured trace with attack traffic e.g., output from nmap scanner → generate flows
  - Drawbacks: too simplistic, network characteristics might not match

- **Shape injection**
  - Inject anomaly of certain shape, e.g., rectangle, directly into metric used by detector
  - Drawbacks: too specific, required for each metric, unrealistic

- **Simulation/Emulation**
  - Emulate network nodes and generate traffic synthetically
  - Drawbacks: limited size of experiments, generation of background traffic is difficult
Anomaly Injection for ADS Testing
Anomaly Injection by Trace Modification

**Advantages**
- High reproducability
- Applicable to different data sets
- Fine-grained anomaly parameterization possible
- Generic with respect to detector since it modifies traffic directly (not derived metrics)

**Disadvantages**
- Interaction with background traffic must be accounted for by the anomaly model
- Attack mitigation approaches cannot be evaluated
- Manual labeling of background traffic still necessary
FLAME: An experimentation framework

<table>
<thead>
<tr>
<th>Helper Functions</th>
<th>Modification Functions</th>
<th>Anomaly Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge</td>
<td>Generate/Add</td>
<td>ingress_shift_del.py</td>
</tr>
<tr>
<td>Read</td>
<td>Delete</td>
<td>tcp_scan_gen.py</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td>syn_flood_gen.py</td>
</tr>
<tr>
<td>Split</td>
<td></td>
<td>syn_flood_del.py</td>
</tr>
</tbody>
</table>

FLAME is flexible, easily extensible, and available (so far upon request)
Setup Example
Flow vs Packet Generation

- Packet-level traffic generation has advantage that flow export settings (timeouts, sampling rates) can be adapted to the underlying trace → less injection artifacts
- Packet-level models might not be available for all attacks (especially if models are extracted from flow traces), potentially more expensive
- FLAME supports packet-level and flow-level traffic generation
Generator/Add Component
Implementation Summary

- Communication between components via named pipes
- Producer-consumer synchronization with bounded buffer size (consumer waits for input from producer)
- Generic flow forwarder interface between core components (based on NetFlow v5)
- Configuration for deleter and packet generator via embedded Python (plug-ins)
- Performance: 140,000 flow records per second (4-way Linux, dual-core CPUs, 2.2 GHz, 8 GB RAM)
Types of Anomalies

- **Subtractive Anomalies**
  - **Delete** flows with defined characteristics from existing traffic
  - Examples are outage events, ingress shifts

- **Additive Anomalies**
  - **Add** flows with defined characteristics to existing traffic
  - Examples are alpha flows, scans, bots

- **Interactive Anomalies**
  - **Delete** existing flows, and **add** new flows
  - Examples are denial of service attacks
Example 1: Ingress Shift

- **Deleter Model (python pseudo code):**
  ```python
  if (start < time < end) {
    if (source or destination IP address within range){
      delete flow;
    }
  }
  ```

- **Deleter Parameters:**
  start, end, shifted IP address range
Example 2: Constant Rate TCP SYN Scan

- **Generator Model on packet level (python pseudo code):**
  ```python
  while (start < time < end) {
    generate TCP SYN packet header
    generate reply with constant delay plus random offset, reply is either nothing, TCP RST, TCP SYN/ACK, or ICMP dest unreachable (source is router)
    advance time by 1 / scan rate
  }
  ```

- **Generator Parameters:**
  - start, end, scan rate, scanned IP address range, scanner's source IP address, probability for each reply type, source/destination port
Example 3: Constant Rate SYN flooding

- **Generator Model and Parameters:**
  - Generate TCP SYN packet at flooding rate, generate reply with certain probability and constant delay plus random offset
  - start, end, flooding rate, reply probability, victim IP address

- **Deleter Model (accounts for loss of replies) and Parameters:**
  - Delete each flow with certain probability if source is victim
  - start, end, IP address of victim, probability for a loss
Plots: Injection of TCP SYN Scan

- Injection in flow trace captured from border router of Swiss educational backbone network (SWITCH AS 559)
- Scanning source internal, scanned destinations external
- Impact on common detection metrics (outgoing traffic):
  - Left: number of flows, right: destination IP address entropy
Conclusion

- **Contributions**
  - Flexible tool for anomaly injection (scripted model plugins)
  - Extensibility: components can be easily added
  - Three example anomalies (ingress shift, ddos, network scan)

- **Future work**
  - Concentrate on model development
  - Evaluate the model accuracy with flow traces
Questions

FLAME is available from
brauckhoff@tik.ee.ethz.ch